PRS muon meeting

CERN, 23 October 2001

New MuonTrajectoryBuilder

Stefano Lacaprara,*INFN, Padova.*

OUTLINE:

- New organisation of MuonTrackFinder,
- Details on new MuonTrajectoryBuilder,
- Performances,
- Future developments.

New Organisation of MuonTrackFinder:

- Internal seed generation moved to independent package:

  MuonSeedGenerator: no mods in the code;
- MuonTrajectoryBuilder split in several classes;
- removal of all “external function” from MuonTrackFinder;
- New classes:
  - ◇ MuonTrajectoryBuilder *is a* TrajectoryBuilder;
  - ◇ MuonTrajectoryUpdator *will be a* “TrajectoryUpdator”;
  - ◇ MuonFTSRefiner;
  - ◇ MuonRecHitMatcher;
  - ◇ MuonBestMeasurement;

Old (monolithic) MuonTrajectoryBuilder:

- Everything is done in the `trajectories` method with the help of external function;
  1. "forward filtering": takes the FTS from the seed, build a sets of reachable layers, use the RecHits in these layers to improve the FTS, **without** building a trajectory. The output is a refined FTS at the outermost reachable layer with a RecHit;
  2. extrapolate by 50 cm outward;
  3. "backward filtering": the true building of a trajectory. Navigation is inward, start from the refined FTS, update trajectory with compatible RecHits;
  4. trajectory saved if two RecHits used; if not:

5. second try of backward filtering: start from a FTS improved but w/o the outermost RecHit, and build a trajectory with a looser chi2 cut;
6. trajectory saved if two RecHits used; if not:
7. third try with 2 RecHits only: build a trajectory from any pair of RecHit in layers reachable from the initial FTS and used the one with best chi2 (if any).

New implementation:

- Main idea is to separate responsibilities of various reconstruction phases into separate classes. The MuonTrajectoryBuilder itself just “use” these classes, define the logic (mainly the navigation) and decide to save or reject a trajectory;
- No big mods in algorithmic part, but an improved framework;
- Still works to do to clean-up the code, avoid duplication, improve interface;
- Better debug available

  `Muon:MuonTrackFinder:debug = 0(no),5(minimal)÷1(max debug)`

“forward filtering”:

- Done by new class `MuonFTSRefiner`;
  - ◇ as before get the reachable layers for the input FTS via MuonNavigableLayer “global” navigation;
  - ◇ get TrajectoryMeasurements in the layers;
  - ◇ select the best one according to chi2 (new class `MuonBestMeasurement`);
  - ◇ the algorithmic part is the same: code duplication, cleanup;
- the improved FTS is not extrapolated to virtual outer surface (**new**): it’s defined at the outermost layer with compatible measurement, with a very loose chi2 cut. If the refiner do not find anything else, no chances that the backward filtering can do it;

“Backward filtering”:

- first get the reachable layers form the refined FTS (**new**);
- get all TrajectoryMeasurements from reachable layers compatible with FTS;
- get best TrajectoryMeasurement from the ones available (MuonBestMeasurement);
- update the trajectory with this TrajectoryMeasurement(s) (new class `MuonTrajectoryUpdator`);
- the TrajectoryBuilder counts the chamber used divided by type (DT,CSC, RPC);
- The Trajectory is saved if two chamber used, where at least one is DT or CSC **new**;

MuonTrajectoryUpdator:

- takes into account the “multi hits” structure of CSC MuEndSegment;
- loops trough the RecHits components (if any);
- propagates the predicted state (defined at the MuEndChamber surface) to the MuEndLayer surface where the MuEndRecHit sits;
- update the Trajectory with the RecHit;
- return the updated Trajectory;
- A base class should sits in CommonDet to define a common interface;
- the interface chosen is probably not the best one:

```
bool MuonTrajectoryUpdator::update(TM* meas, Trajectory& traj);
```

  return true if at least one RecHit is used: the updated Trajectory is “traj”;
- Need to clean up the algorithm;

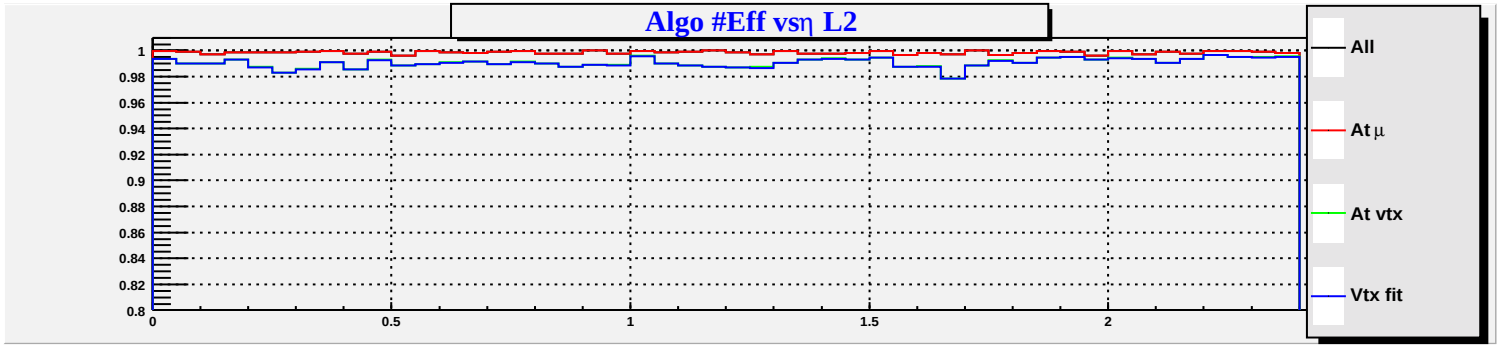
Second and third try:

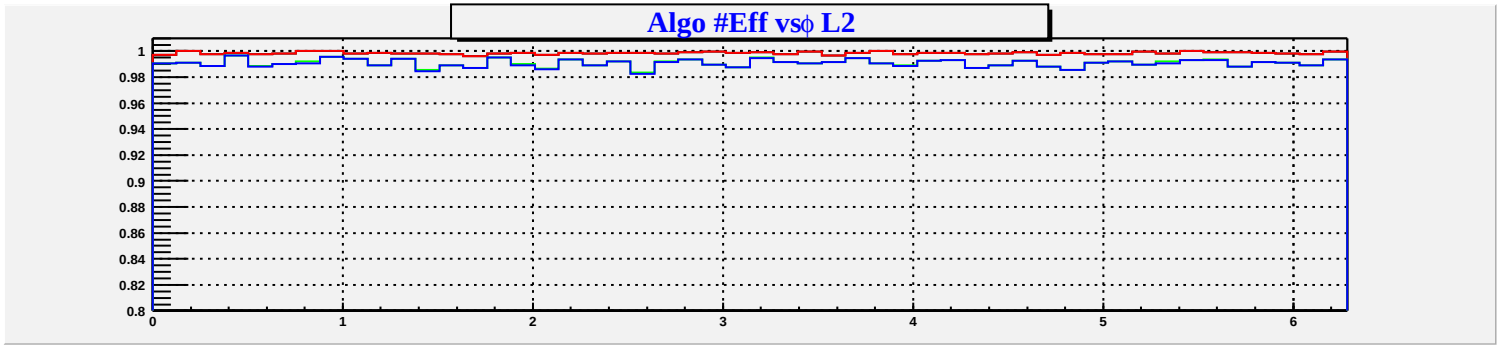
- The second try follow the same steps, but:
  - it's done only if the number of chambers used in the MuonFTSRefiner is at least two;
  - the initial FTS is the one delivered by MuonFTSRefiner without using the outermost layer;
  - the chi2 cut to collect TrajectoryMeasurements is increased;
- if everything fails, the try to build a trajectory from two chamber only: new class MuonRecHitMatcher;

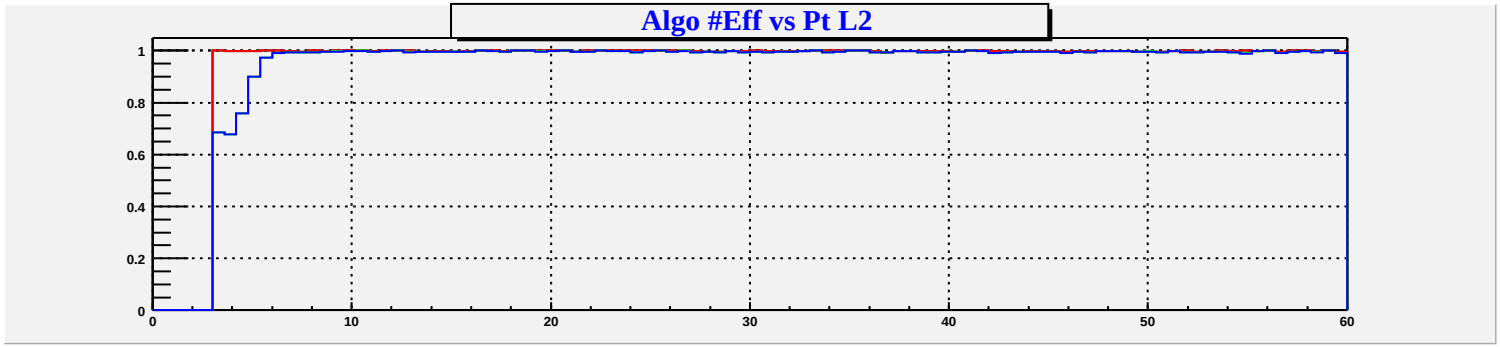
MuonRecHitMatcher:

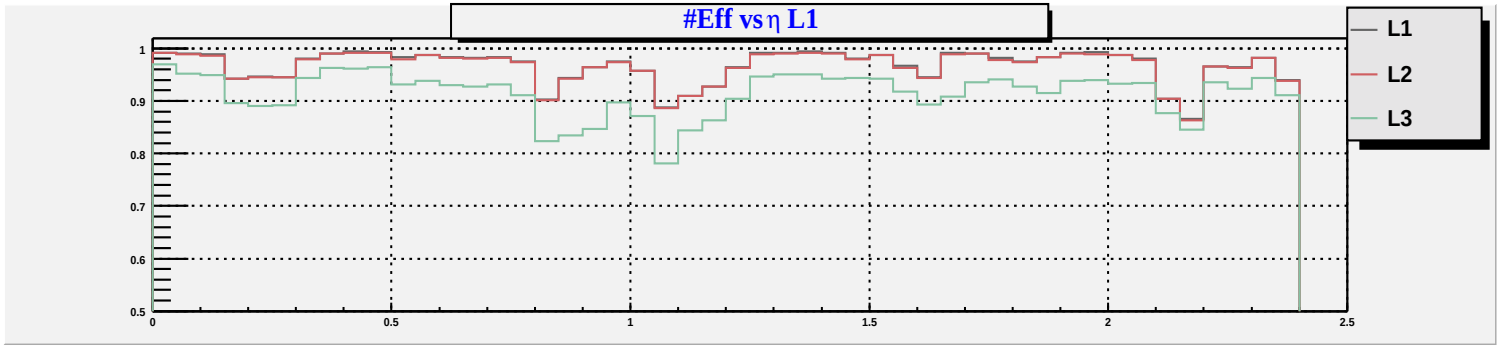
- Was external function RecHitMatcher (very expensive loop trough all possible layers pair);
  - loops only trough the layers where the refiner found some RecHits (optimisation!);
  - build a Trajectory out of all pair of RecHits compatible with seed FTS;
  - return the one with the best chi2;
  - also here code duplication: need clean-up;

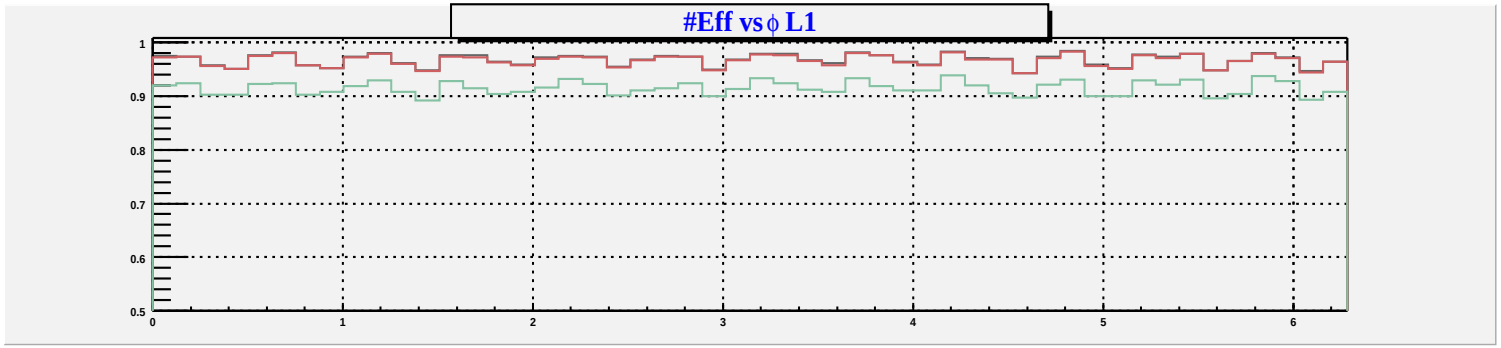
Performances on Single muon flat sample:

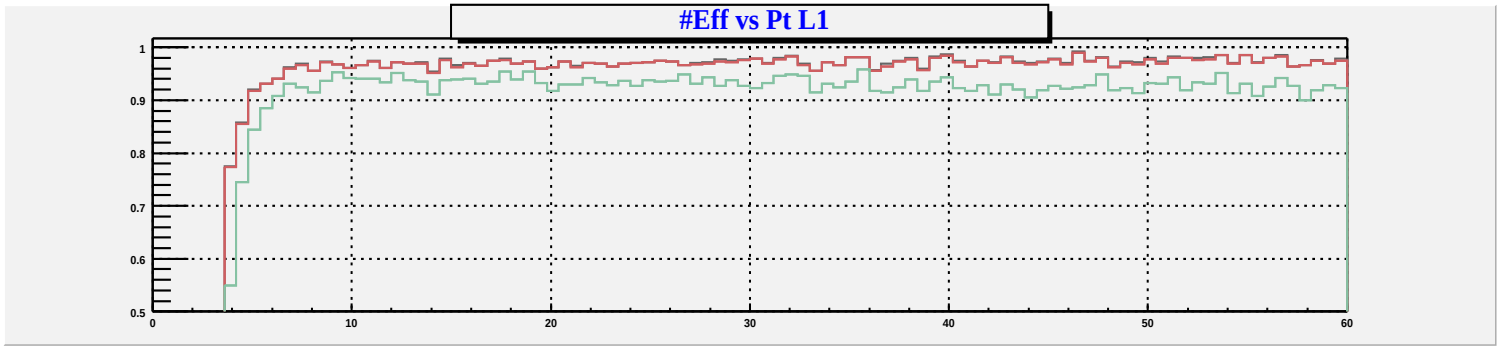




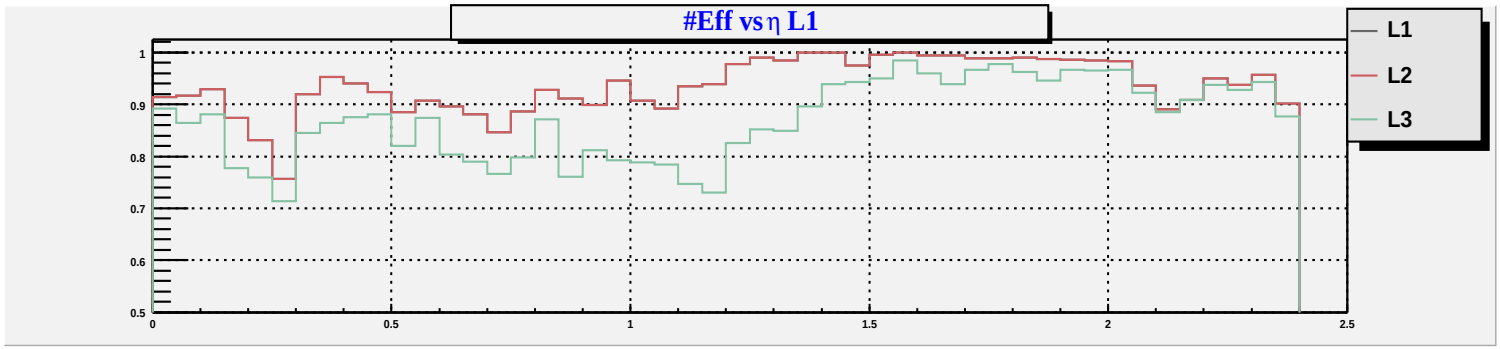


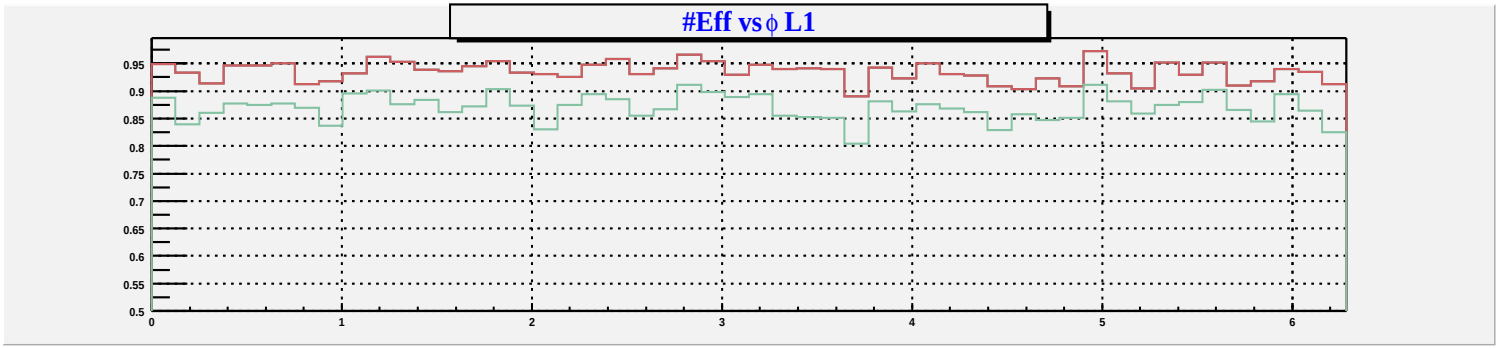

#Eff vs η L1
L1
L2
L3
1
0.9
0.8
0.7
0.6
0.5
0
0.5
1
1.5
2
2.5

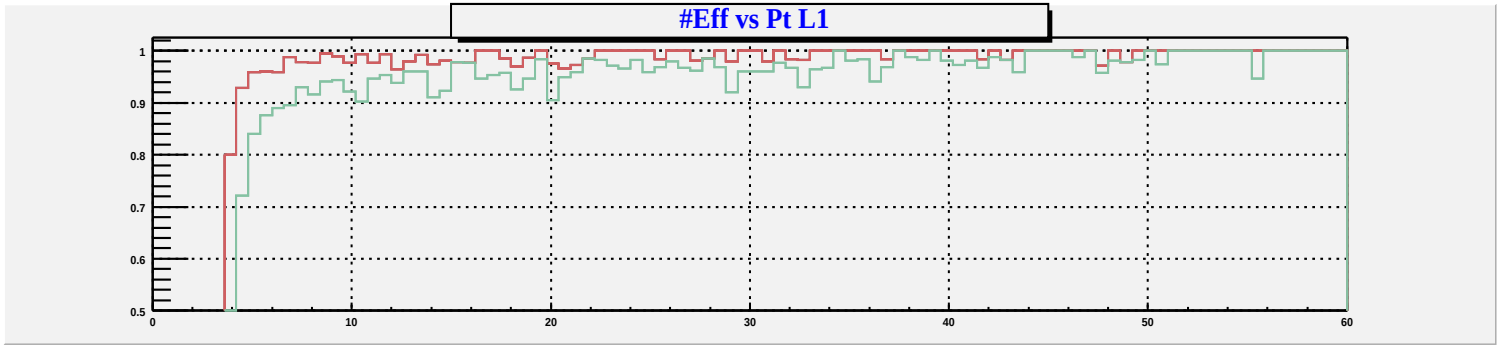

#Eff vs ϕ L1
1
0.9
0.8
0.7
0.6
0.5
0
1
2
3
4
5
6


#Eff vs Pt L1
1
0.9
0.8
0.7
0.6
0.5
0
10
20
30
40
50
60

Performances on mu_realZ sample:







General comments and future development:

- MuonTrajectoryBuilder is now much more readble, the complexity has been moved to the auxiliary classes;
- It should be easy to try to plug-in the new “step by step” Navigation and see how it works (I remind you: no RPC yet!!);
- Code duplication: MuonTrajectoryUpdator used everywhere (not only in MuonTrajectoryBuilder);
- CommonDet interface for the TrajectoryUpdator;
- Cut definition in .orcarc (chi2, no. of chamber used, etc);
- Make the number of chamber used of each kind (DT,CSC, RPC) available to the end-user (could help in defining the quality of a MuonTrack);

- Documentation: I know, I know... (already some improvement: a lot of classes has been doxygenate);
- The chi2 cut we apply are really huge!!! Do we need it?? Do we understand the error in the Rechits ?? (answer is NO!);
- Use of RCP now is default: mature enough.
- Two more helper classes MuBar(End)SegmentPrinter available in MBClusterData and MuEndDetector to easy dump of segment information (down to Hit level).

Timing consideration:

- The TrejectoryBuilder is slow;
- almost all time spent in propagation;
- most propagation are done to find out the right DetUnit inside a given DetLayer: we must optimise this part for the Muon system (not critical for the Tracker);
- other smaller optimisation possible (e.g. use GtfPropagator inside MuEndChamber instead of Geane) but useless. For a couple of events (I don't remember how mutch ~ 10) using Gtf improve time spent from 0.120 s to 0.020 s, but other ~ 30 s spent elsewhere!